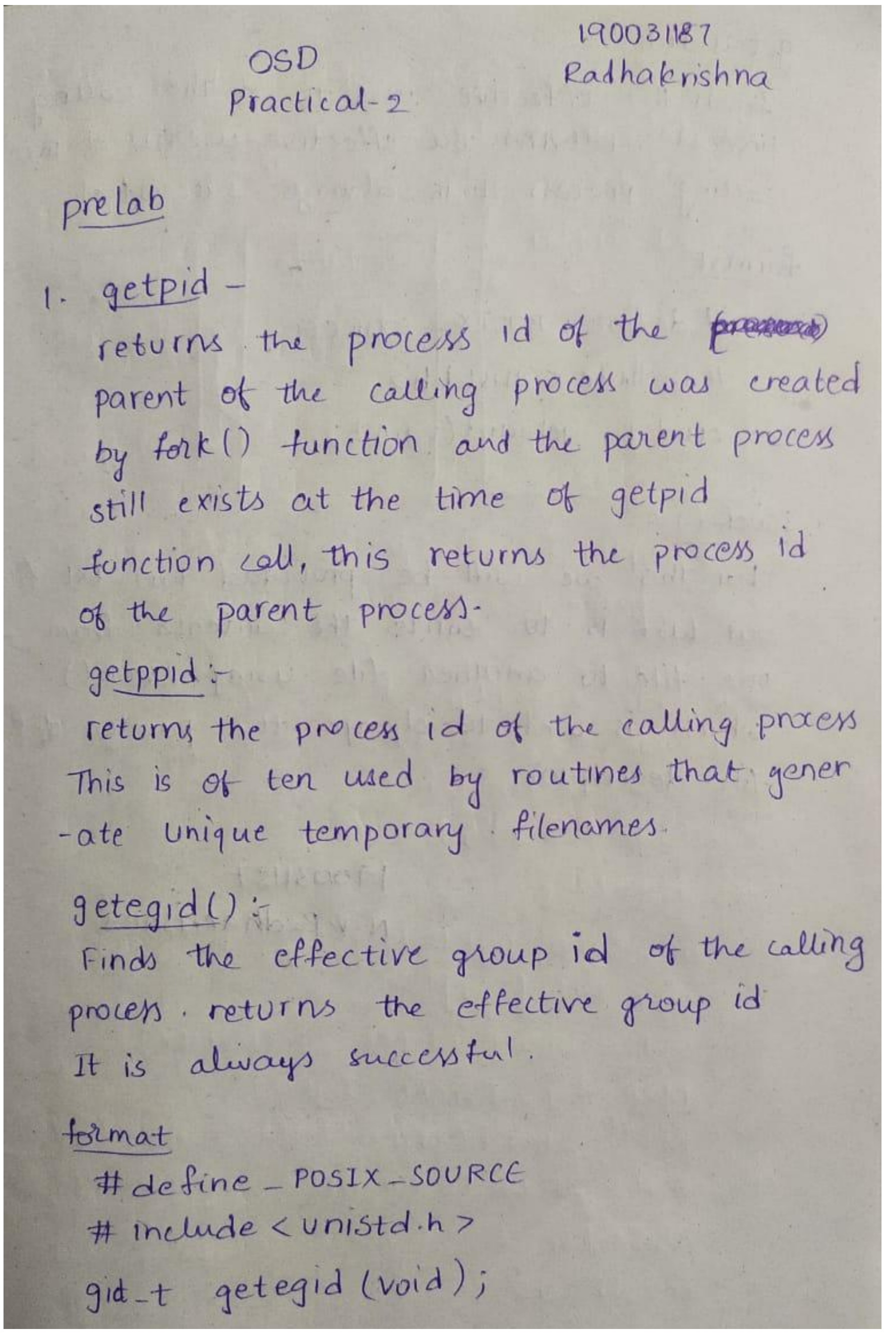
Operating System Design – 19CS2106S

Lab Experiment - 2

Pre-Lab 2:

- Write the description for the following
  getpid: Return the current process's pid
  getppid: Obtains a parent process's ID number getuid and getgid get the user or group ID of the process, respectively.
  getegid- get the effective group ID
  geteuid - get the effective user ID errno, a global variable that holds the numeric code of the last system call error perror (), a subroutine that describes system call errors.
  Moving in a File: lseek () dup:Duplicate fd



OSD
Practical-2

190031187
Radhakrishna

prelab

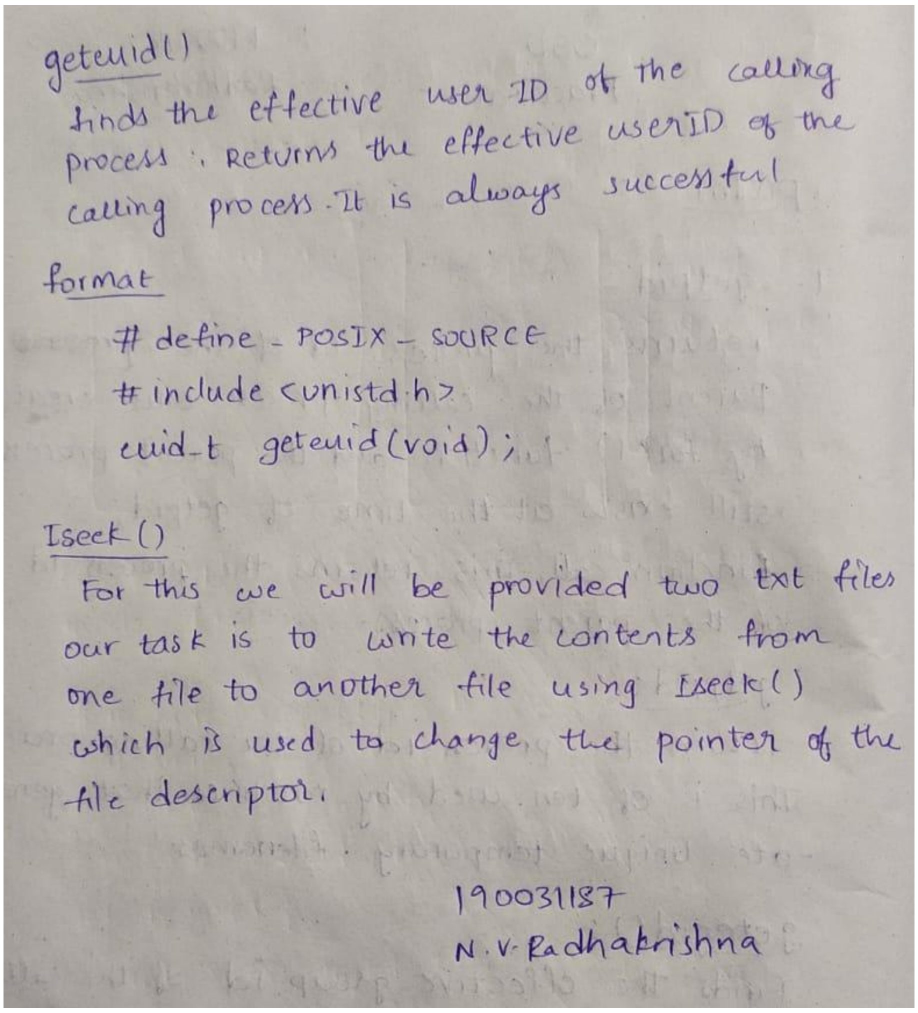
1. getpid –
returns the process id of the ~~(process)~~ parent of the calling process was created by fork() function and the parent process still exists at the time of getpid function call, this returns the process id of the parent process.

getppid :-
returns the process id of the calling process This is of ten used by routines that gener -ate unique temporary filenames.

getegid() :-
Finds the effective group id of the calling process. returns the effective group id It is always successful.

format

```
#define _POSIX_SOURCE
# include <unistd.h>
gid_t getegid (void);
```



geteuid()

finds the effective user ID of the calling process : Returns the effective userID of the calling process. It is always successful

format

```
# define - POSIX - SOURCE
# include <unistd.h>
euid_t geteuid(void);
```

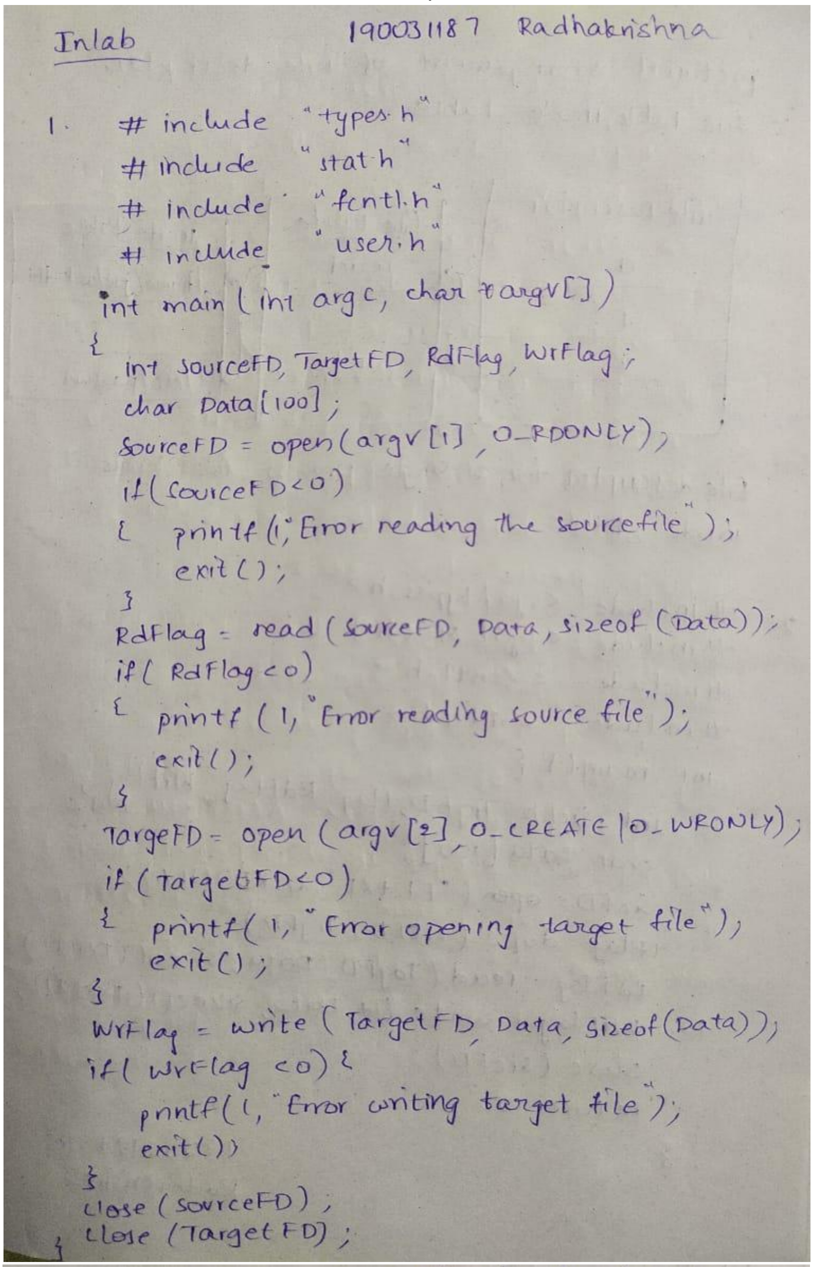
Iseek ()

For this we will be provided two txt files our task is to write the contents from one file to another file using Iseek() which is used to change the pointer of the file descriptor.

190031187
N.V.Radhakrishna

In-Lab 2:

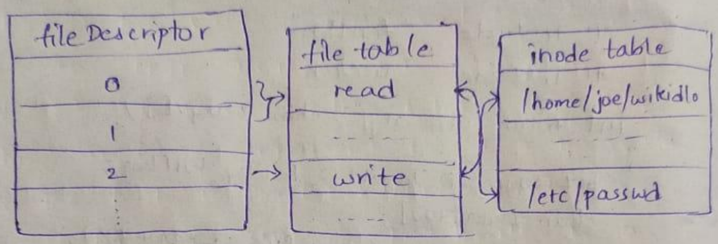
- ccp.c: Copying a File. Show a pictorial arrangement of File Descriptor, File and Inode tables for a single process that has two different files open.



Inlab 190031187 Radhakrishna

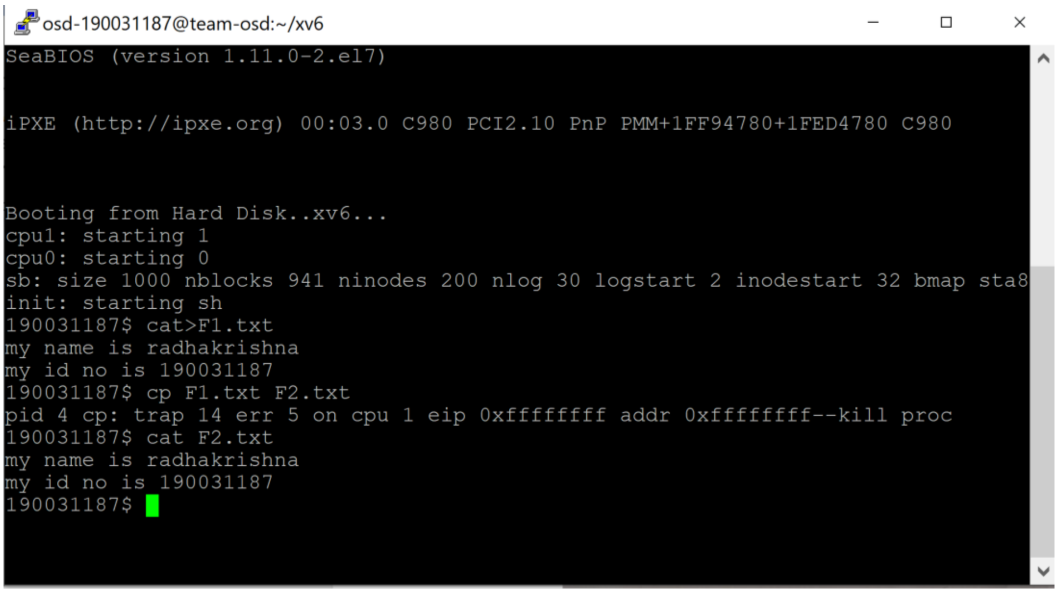
```
1.  # include "types.h"
    # include "stat.h"
    # include "fcntl.h"
    # include "user.h"
   int main ( int argc, char *argv[] )
   {
    int SourceFD, TargetFD, RdFlag, WrFlag ;
    char Data[100] ;
    SourceFD = open(argv[1], O_RDONLY);
    if(SourceFD<0)
    {  printf(1," Error reading the sourcefile" );
       exit();
    }
    RdFlag = read (SourceFD, Data, sizeof (Data));
    if( RdFlag<0)
    {  printf (1, "Error reading source file");
       exit();
    }
   TargeFD = open (argv[2], O_CREATE |O_WRONLY);
   if (TargetFD<0)
   {  printf( 1, "Error opening target file");
      exit() ;
   }
   WrFlag = write ( TargetFD, Data, sizeof(Data));
   if( WrFlag <0) {
      printf(1, "Error writing target file");
      exit())
   }
   close ( sourceFD) ;
   close (TargetFD) ;
  }
```

pictorial Arrangement of File descriptor, file table, inode table



File descriptor for a single process file table & inode table

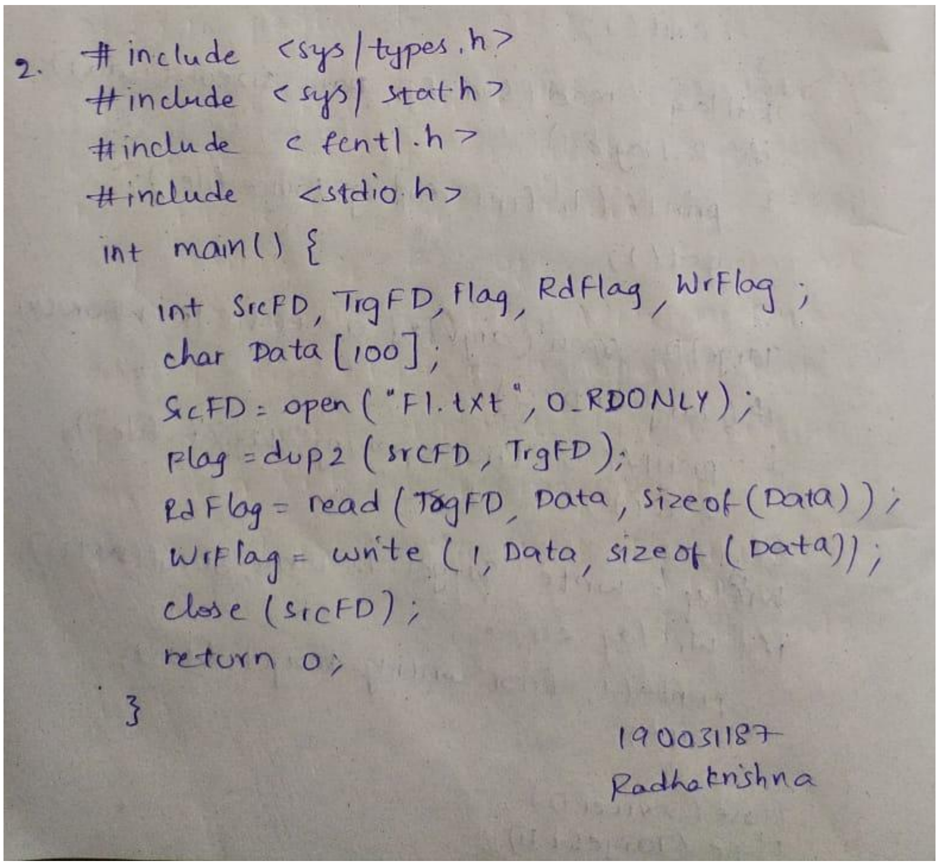
OUTPUT CP.C



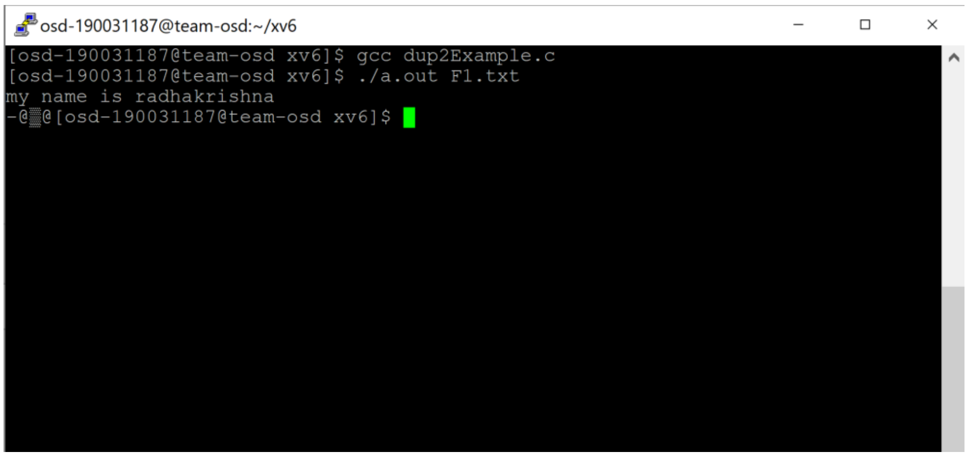
osd-190031187@team-osd:~/xv6

```
SeaBIOS (version 1.11.0-2.el7)


iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+1FF94780+1FED4780 C980



Booting from Hard Disk..xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap sta8
init: starting sh
190031187$ cat>F1.txt
my name is radhakrishna
my id no is 190031187
190031187$ cp F1.txt F2.txt
pid 4 cp: trap 14 err 5 on cpu 1 eip 0xffffffff addr 0xffffffff--kill proc
190031187$ cat F2.txt
my name is radhakrishna
my id no is 190031187
190031187$ 
```

- Write a system program which will opens files in the parent and uses dup2 in the child to reassign the descriptors .



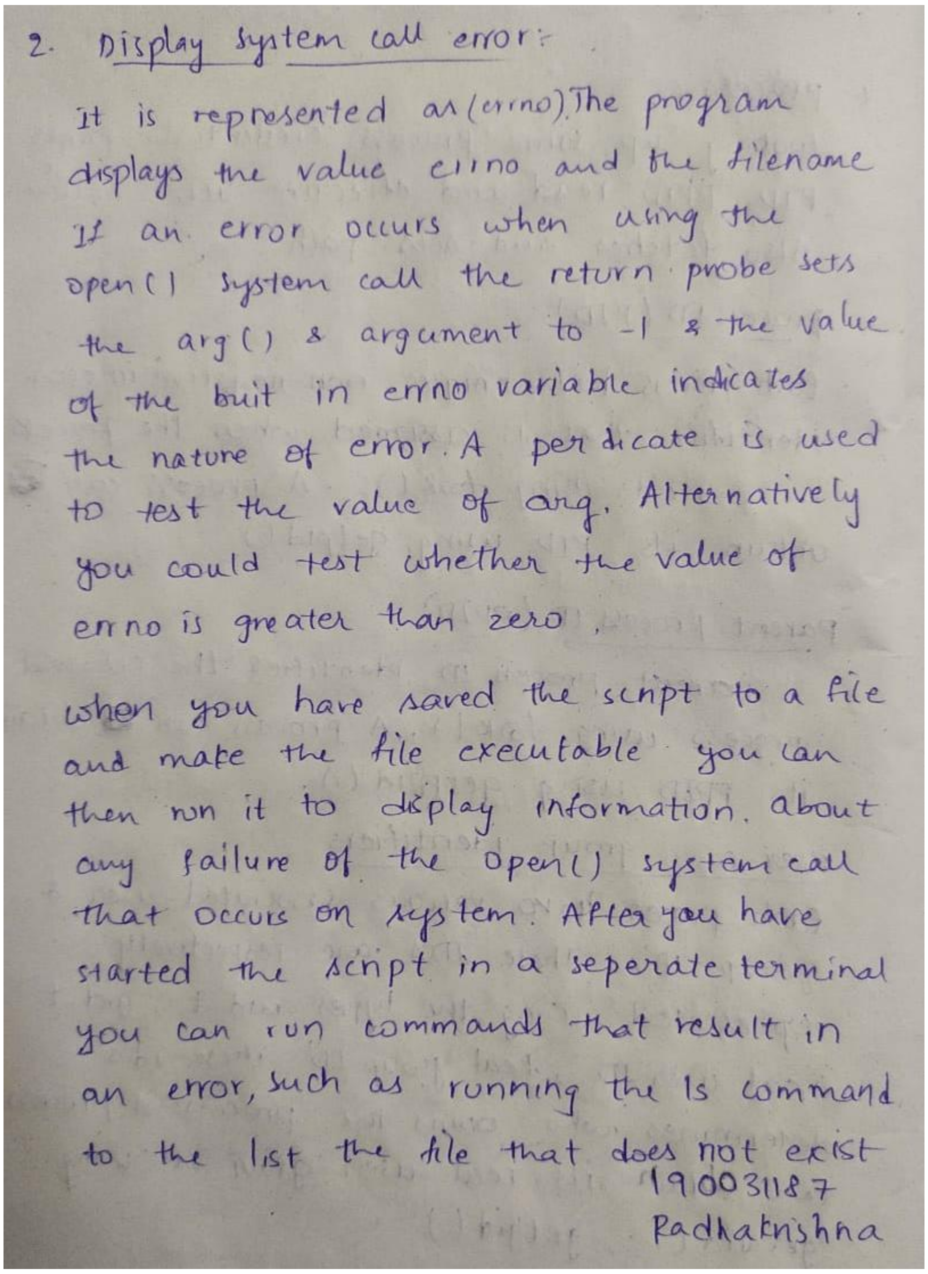
```
2.  #include <sys/types.h>
    #include <sys/stat.h>
    #include < fcntl.h >
    #include <stdio.h>
     int main() {
         int SrcFD, TrgFD, flag, RdFlag, WrFlag ;
         char Data[100];
         ScFD = open ("F1.txt", O_RDONLY);
         Flag = dup2 (srcFD, TrgFD);
         RdFlg = read (TrgFD, Data, sizeof (Data));
         WrFlag = write (1, Data, sizeof (Data));
         close (srcFD);
         return 0;
     }
                                    190031187
                                    Radhakrishna
```


osd-190031187@team-osd:~/xv6

```
[osd-190031187@team-osd xv6]$ gcc dup2Example.c
[osd-190031187@team-osd xv6]$ ./a.out F1.txt
my name is radhakrishna
-@▒@[osd-190031187@team-osd xv6]$
```

Post-Lab 2:

- show_error s.c: Displaying system call errors with perror Filename provided as argument

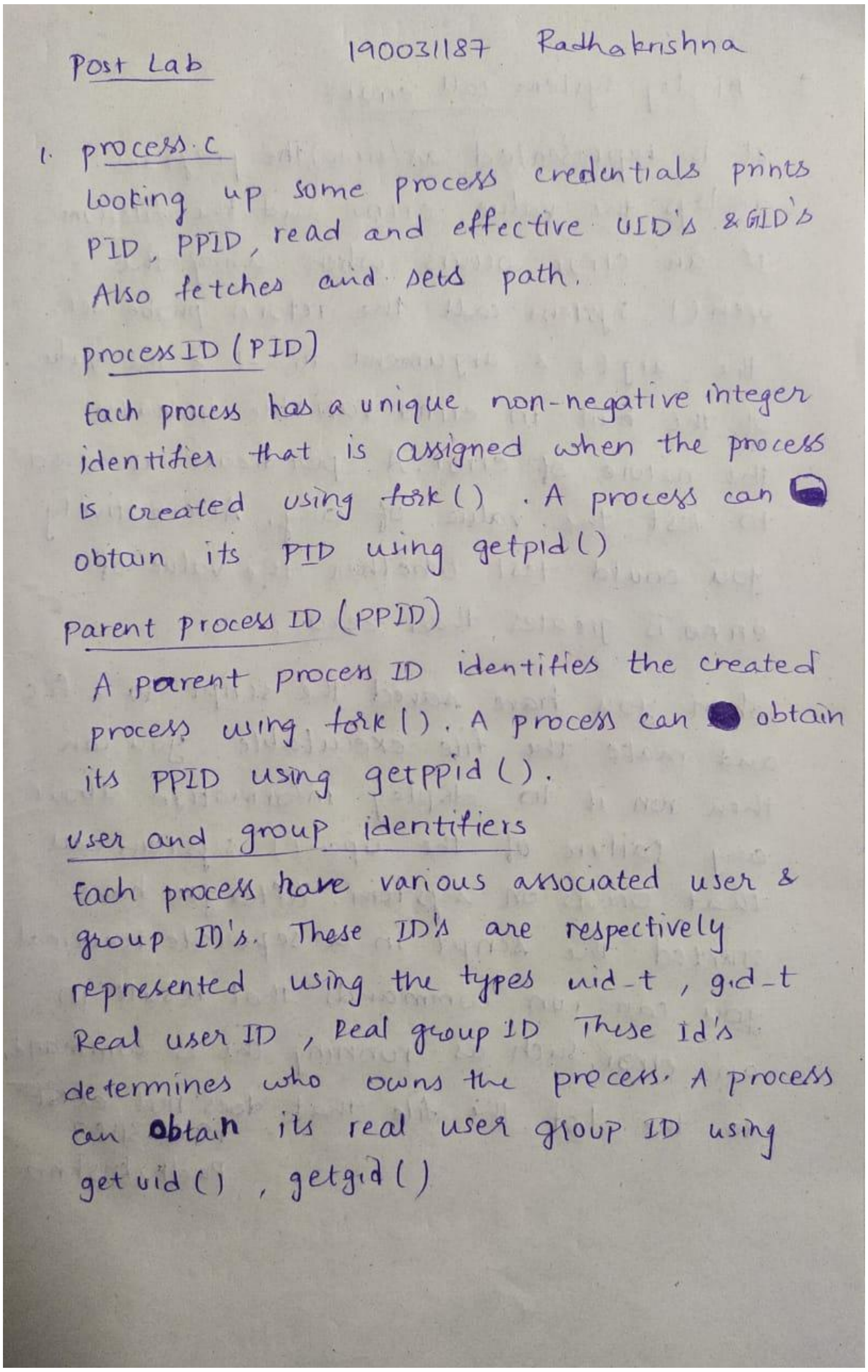

2. Display system call error:

It is represented as (errno) The program displays the value errno and the filename If an error occurs when using the open() system call the return probe sets the arg() & argument to -1 & the value of the buit in errno variable indicates the nature of error. A perdicate is used to test the value of arg. Alternatively you could test whether the value of errno is greater than zero.

when you have saved the scirpt to a file and make the file executable you can then run it to display information. about any failure of the open() system call that occurs on system. After you have started the scirpt in a seperate terminal you can run commands that result in an error, such as running the ls command to the list the file that does not exist

190031187
Radhakrishna

- process.c: Looking Up Some Process Credentials Prints PID, PPID, real and effective UIDs and GIDs Also fetches and sets PATH.



Post Lab 190031187 Radhakrishna

1. process.c
Looking up some process credentials prints PID, PPID, read and effective UID's & GID's Also fetches and sets path.

processID (PID)
Each process has a unique non-negative integer identifier that is assigned when the process is created using fork() . A process can obtain its PID using getpid()

Parent process ID (PPID)
A parent process ID identifies the created process using fork(). A process can obtain its PPID using getppid().

User and group identifiers
Each process have various associated user & group ID's. These ID's are respectively represented using the types uid_t , gid_t
Real user ID , Real group ID These Id's determines who owns the process. A process can obtain its real user group ID using getuid() , getgid()